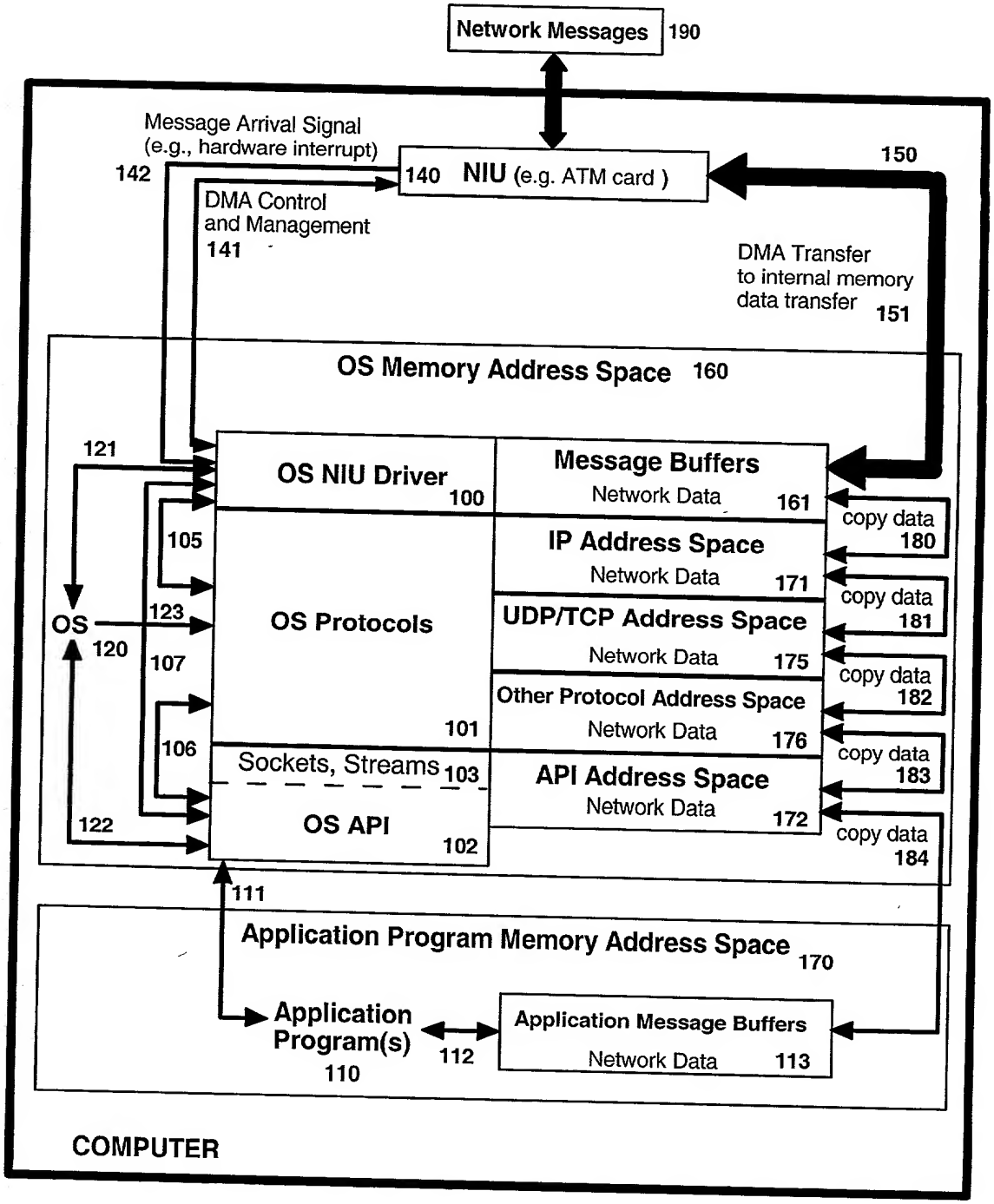

Figure 1. Overview of prior Art Network Communication System

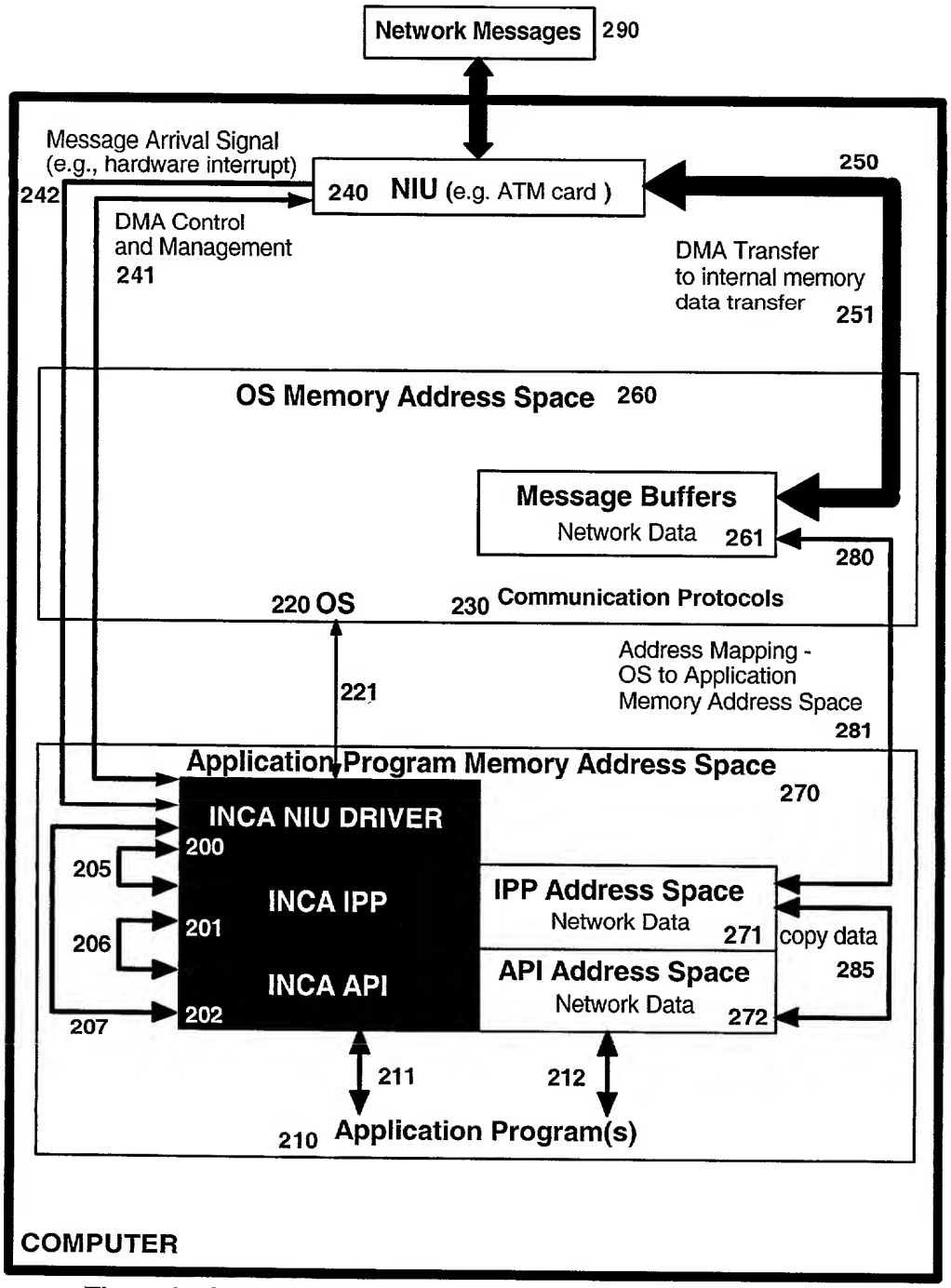

Figure 2. Overview of the INCA Network Communication System (INCA Integrated into Application)

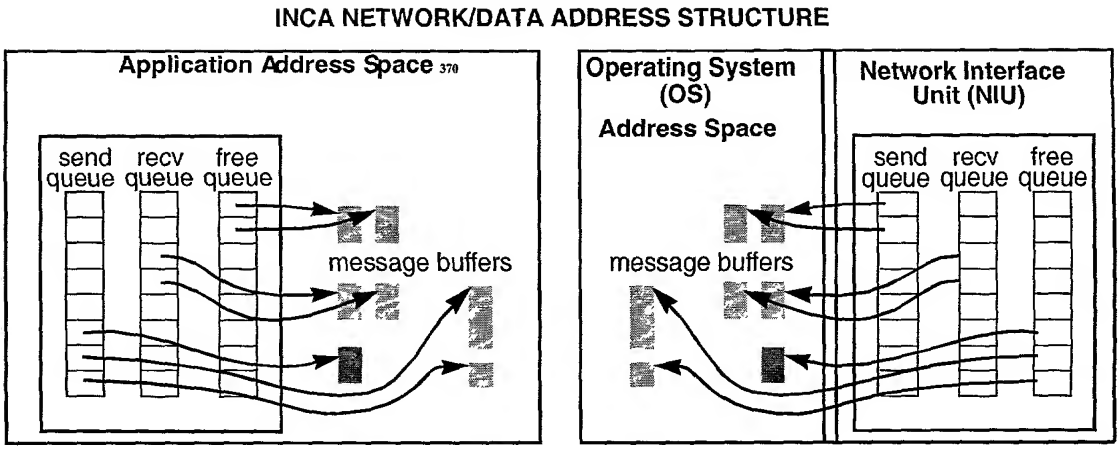

Figure 3. INCA Network Data/Mapping Address Data Structure Mechanism

```
for( i = 0; i < 1000; i++ )
    msgData[i]++;                    /* LOAD, ADD, STORE */
for( i = 0; i < 1000; i++ )
    msgData[i] = ~msgData[i];        /* LOAD, COMPLEMENT, STORE */
```

Figure 4a. Two non-IPP For-Loops Examples for typical prior Art multiple Protocol processing Result in a Read (load) and a Write (store) for each Protocol's individual Loop

```
for( i = 0; i < 1000; i++ ){
    temp = msgData[i];               /* LOAD */
    temp++;                          /* ADD */
    temp = ~temp;                    /* COMPLEMENT */
    msgData[i] = temp;               /* STORE */
            }
```

Figure 4b. Examples of INCA's Integrated IPP For-Loops for multiple Protocol processing result in one read (load) and a write (store) for all processed Protocols

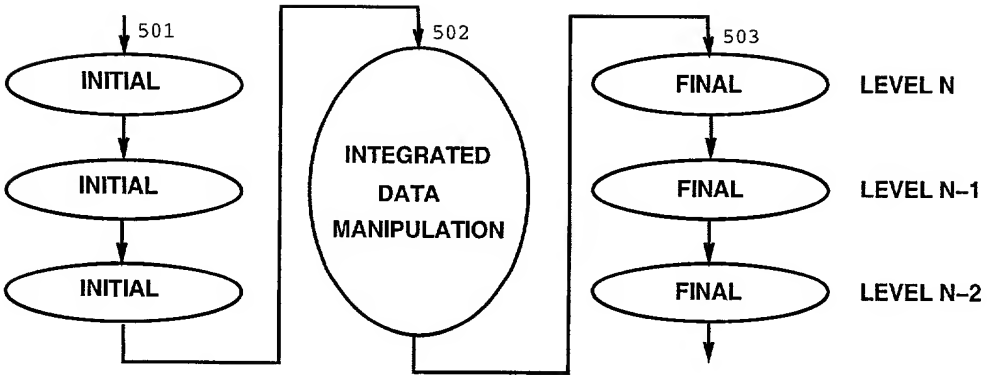

Figure 5. INCA Integrated Protocol Processing (IPP) Execution Stages

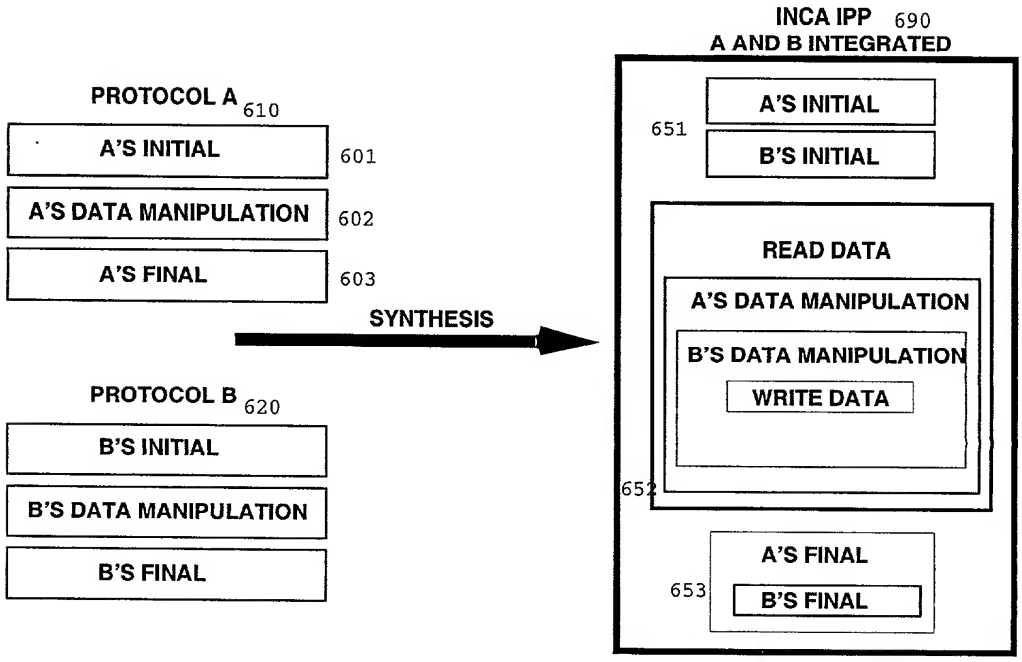

Figure 6. INCA's IPP Method of integrating multiple Protocols

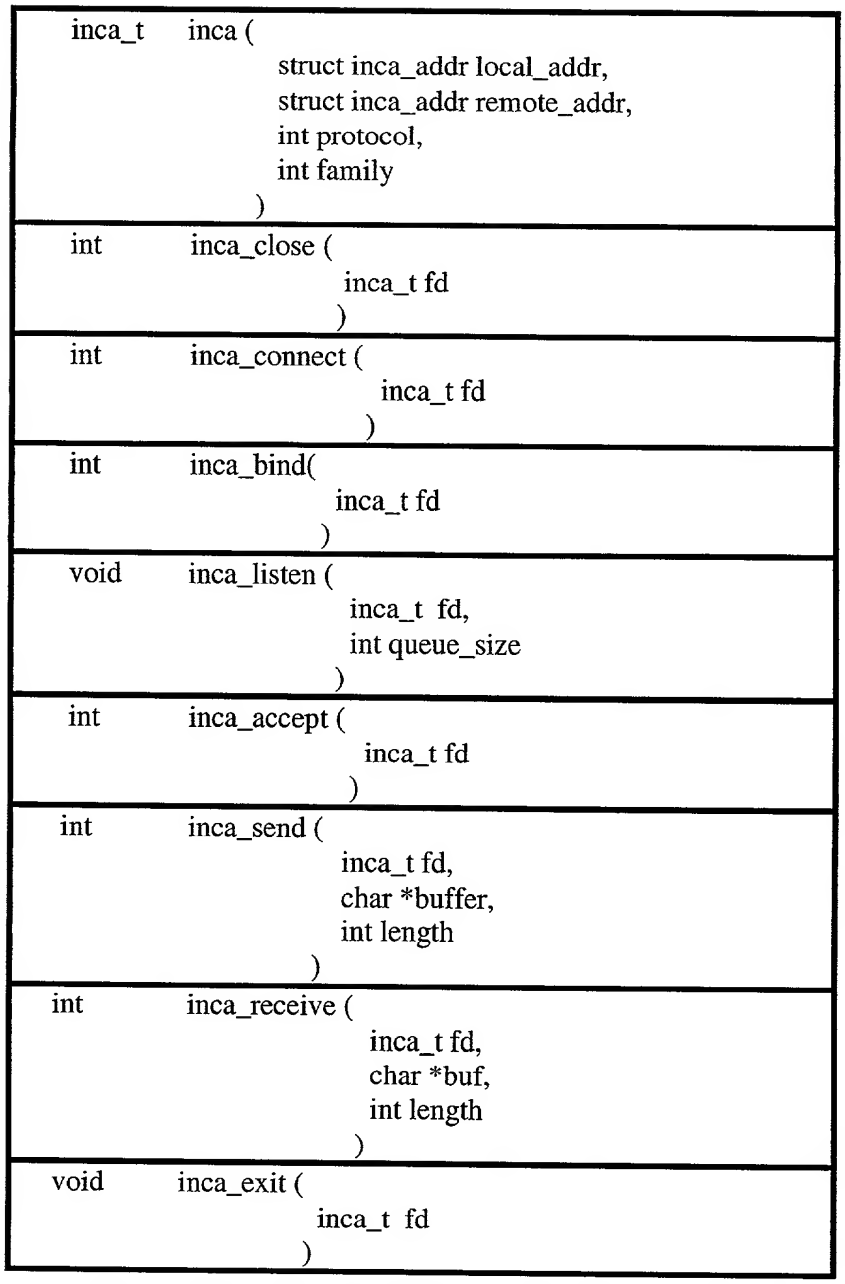

Figure 7. The INCA API Calls and Call Parameters

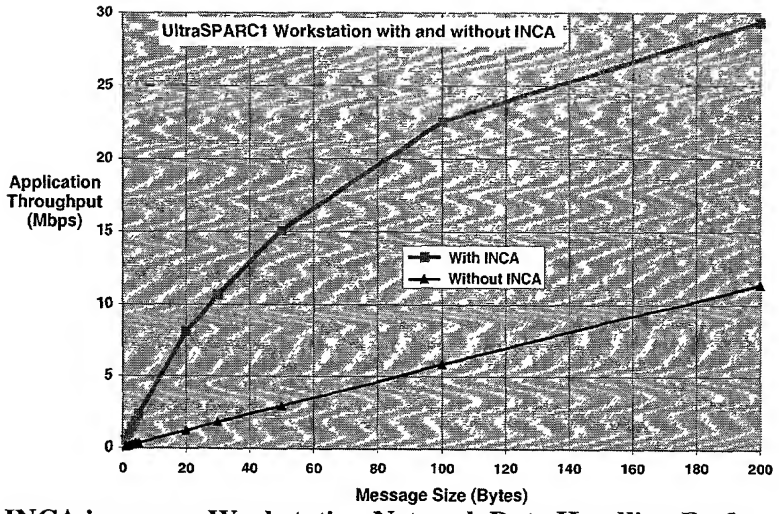

Figure 8. INCA improves Workstation Network Data Handling Performance 260-760%

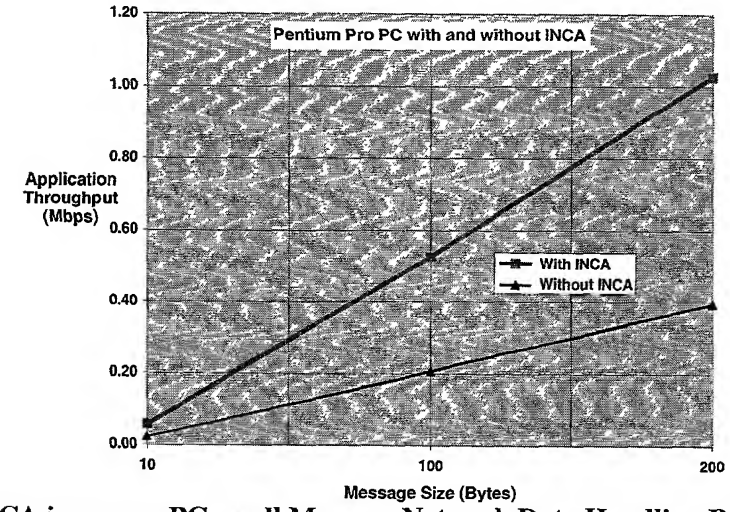

Figure 9. INCA improves PC small Message Network Data Handling Performance by 260-275%

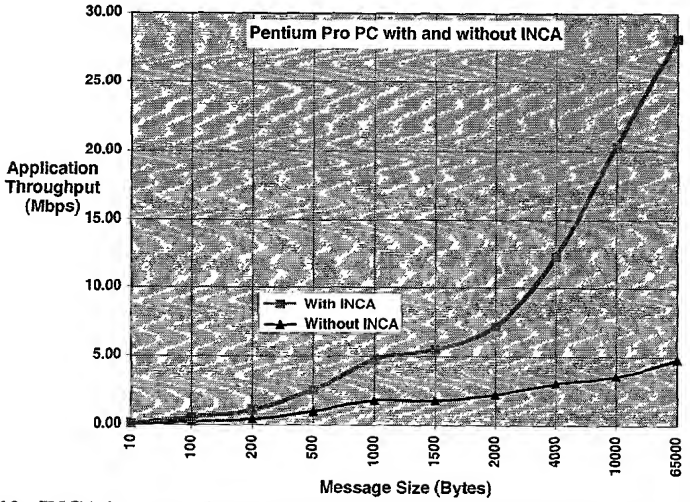

Figure 10. INCA improves PC Network Data Handling Performance up to 590%

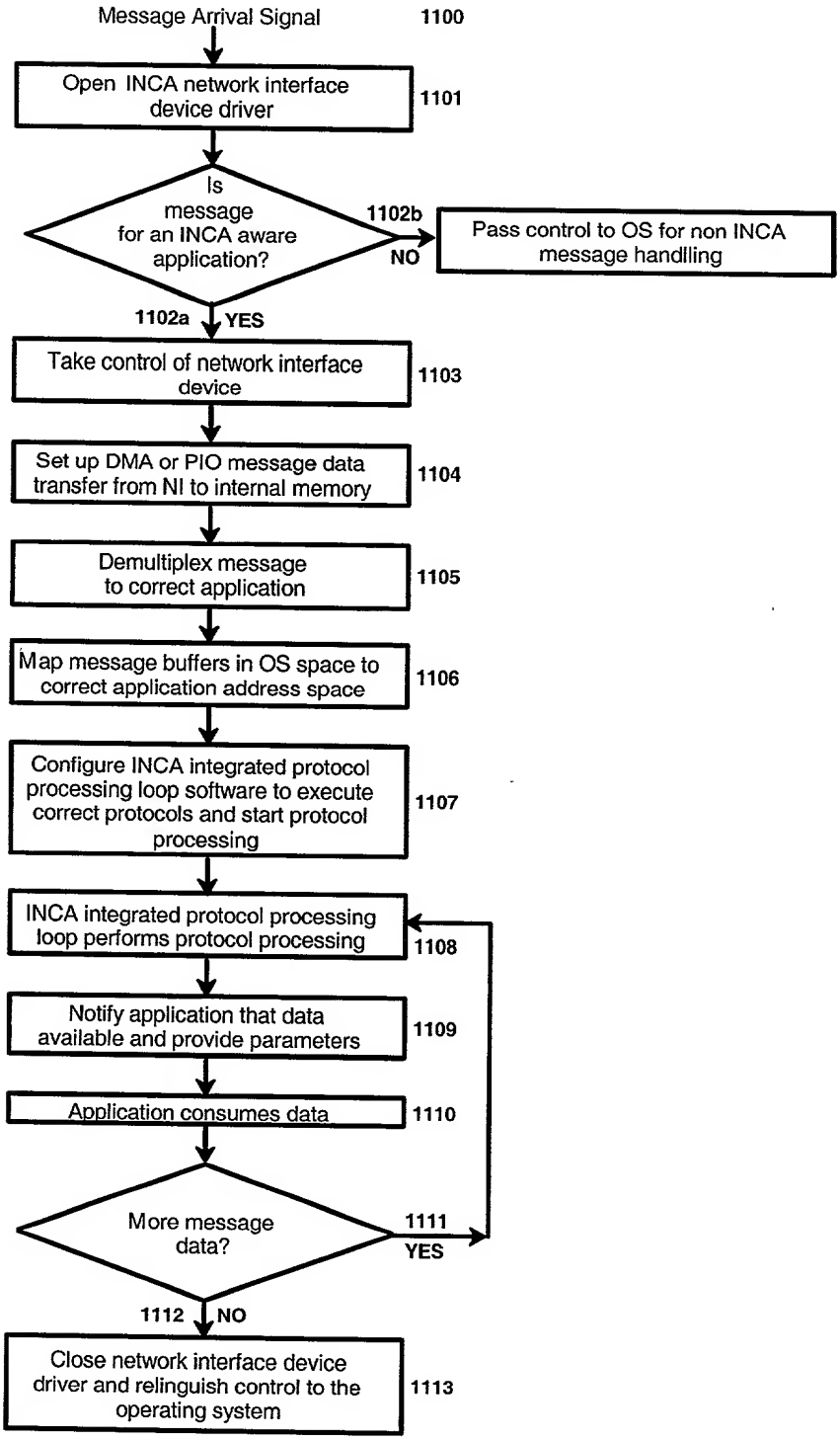

Figure 11. INCA's Management and Control Flow

```
/* Character/block device ops for INCA Network Driver*/
static struct cb_ops inca_cb_ops = {
                        inca_open,      /* Device Open          */
                        inca_close,     /* Device Close         */
                        nodev,          /* strategy     */
                        nodev,          /* print        */
                        nodev,          /* dump                 */
                        nodev,          /* read                 */
                        nodev,          /* write                */
                        inca_ioctl,             /* ioctl                */
                        nodev,                  /* devmap       */
                        inca_mmap,              /* mmap                 */
                        ddi_segmap,             /* segment map          */
                        nochpoll,
                        nodev,                  /* prop_op      */
                        NULL,                   /* streams      */
                        (D_NEW|
                              D_MP)                     /*could be D_MP*/
                        };

/* Device operations */
static struct dev_ops inca_ops = {
                                DEVO_REV,
                                0,
                                inca_getinfo,           /* Info */
                                inca_identify,          /* Identify */
                                nulldev,        /* probe        */
                                inca_attach,            /* Device attatch*/
                                inca_detach,            /* Device detatch*/
                                nodev,
                                &inca_cb_ops,       /*Pointer to ops*/
                                (struct bus_ops *)NULL
};
```

Figure 12. INCA Network Driver Entry Points inca_cb_ops Structure

```
for(j=0;j<Length;j+=IPP_UNIUT) {
   /* Read IPP_UNIUT of data, 4/8 byte at a time */
         Data = *input++; / Input Buffer is word aligned */
   if ( IPP_UNIUT == 4) {
         /* Byte Swap */
         Data = ((Data & 0x00FF00FF00) <<8 )|(Data & 0xFF00FF00)>>8);
         /* Check Sum */
         csum += (Data & 0x0000FFFF) + (Data &0xFFFF0000);
   } else {
         /* Byte Swap */
Data = ((Data & 0x00FF00FF00FF00FF) <<8 ) | (Data & 0xFF00FF00FF00FF00)>>8);
         /* Check Sum */
         csum += (Data & 0x000000000000FFFF) + (Data &0x00000000FFFF0000) +
         (Data & 0x0000FFFF00000000) + (Data &0xFFFF000000000000);
   }
```

Figure 13. INCA IPP Example Implementation - Integrating Byte-Swap and Internet Checksumming for 32 and 64 bits

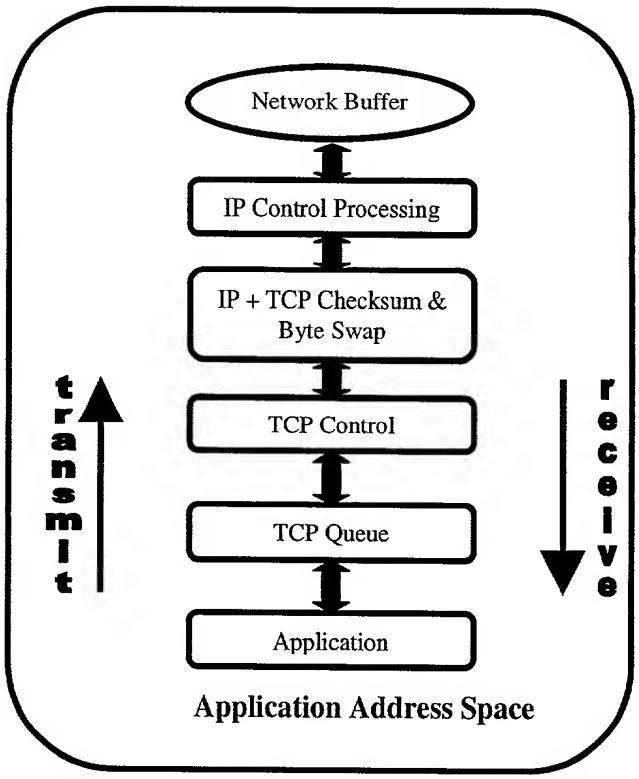

Figure 14. INCA IPP TCP Overview

```
/* Type cast the 4-byte character array of IP address to 4-byte int variable */
if ( *(int *) ip->ip_dst = = *(int *) Connector->ip_src) {
        /* The IP Address is our address. So proceed. */
        register int ip_csum;
        register int udp_csum;

        ip_csum  = *(short *)ip->ip_dst + *(short *)(ip->ip_dst + 2);
        udp_csum =  ip_csum;
        /* Do the rest of the Processing */
}
```

Figure 15. INCA IPP Example of Exploiting Locality with Checksum and Control Processing Integration

```
static int  tcpsend(int tcbnum, Bool rexmt) {
            struct      tcb     *ptcb = &tcbtab[tcbnum];
            struct      ep      *pep;
            struct      ip      *pip;
            struct      tcp     *ptcp;
            char        *pch;
            char *tmp;
            int         i, datalen, tocopy, off, newdata;
            pep = (struct ep *)inca_tx_alloc(sizeof(struct ep));/* Allocate Aligned
                                                                    INCA Memory */
            if (pep == (struct ep *)SYSERR)
                return SYSERR;      /* Allocation Failed */
            pip = (struct ip *)pep->ep_data;    /* Typecast to IP */
           /* INTEGRATING CHECKSUMMING + DATA PROCESSING
              FOR IP AND TCP
           */
            *(int *)pip->ip_src = *(int *)ptcb->tcb_lip;
            ptcb->con->ip_csum += ((*(int *)pip->ip_src & 0xFF00) >> 16) +
                                                    *(int *)pip->ip_src & 0x00FF;
            *(int *)pip->ip_dst = *(int *)ptcb->tcb_rip;
            ptcb->con->ip_csum += ((*(int *)pip->ip_dst & 0xFF00) >> 16) +
                                                    *(int *)pip->ip_dst & 0x00FF;
            ptcp->tcp_sport = ptcb->tcb_lport;
            ptcp->tcp_dport = ptcb->tcb_rport;

            ptcb->con->tcp_csum += ((*(int *)&ptcp->tcp_sport & 0xFF00) >>16) +
                                    *(int *)&ptcp->tcp_sport & 0x00FF;
           /* Continue the TCP send processing */
}
```

Figure 16. INCA IPP Integrating TCP + IP Checksumming with Header Creation for Maximum Locality

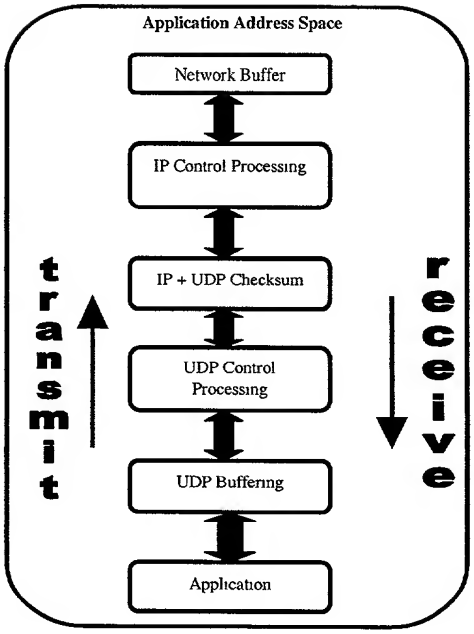

Figure 17. INCA IPP UDP Overview

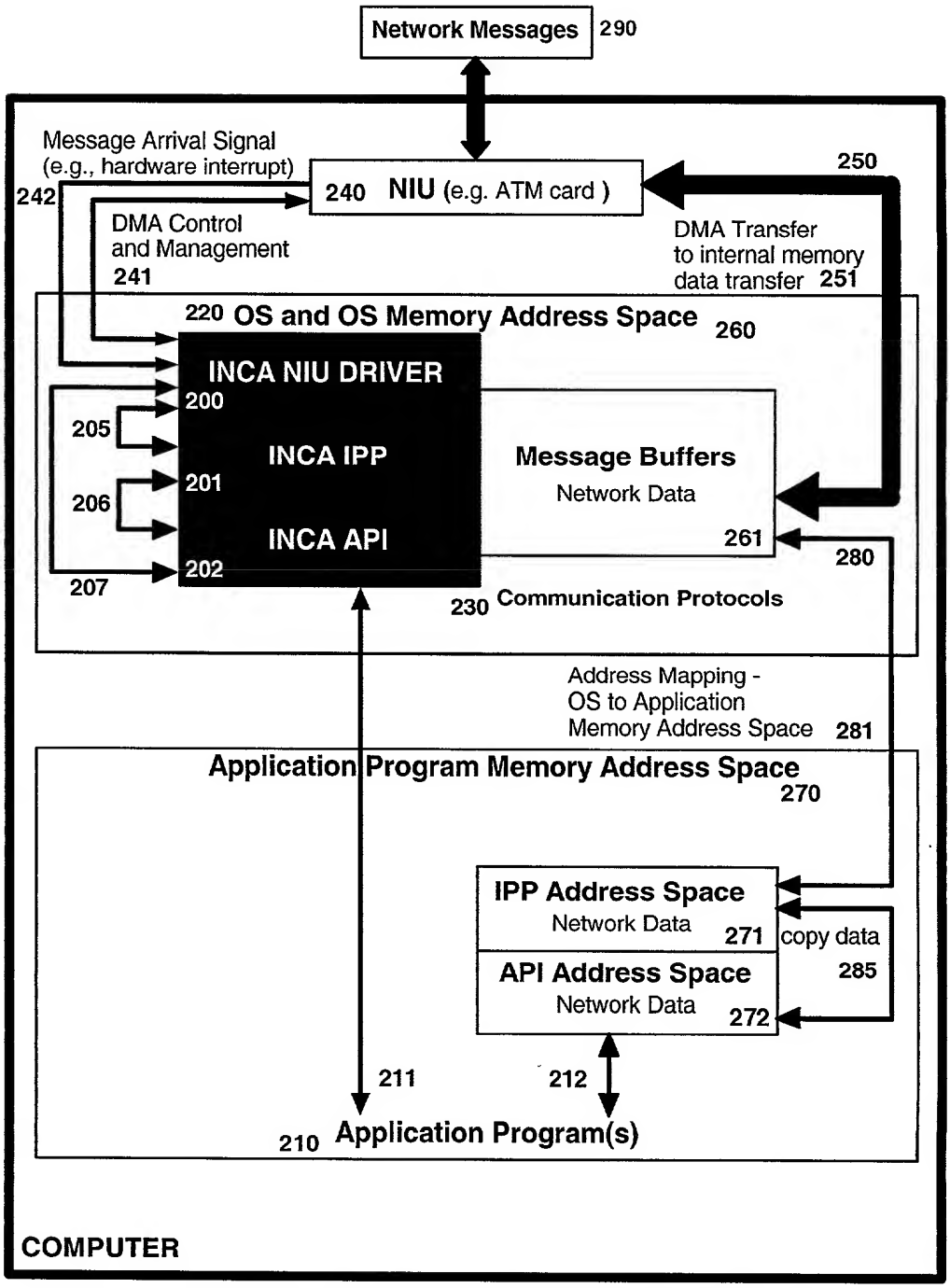

Figure 18. INCA Network Communication System Integrated into the OS